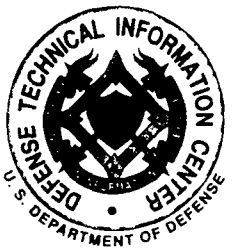

Defense Technical Information Center Compilation Part Notice

This paper is a part of the following report:

• *Title:* Technology Showcase: Integrated Monitoring, Diagnostics and Failure Prevention. Proceedings of a Joint Conference, Mobile, Alabama, April 22-26, 1996.

• *To order the complete compilation report, use:* AD-A325 558

The component part is provided here to allow users access to individually authored sections of proceedings, annals, symposia, etc. However, the component should be considered within the context of the overall compilation report and not as a stand-alone technical report.

SYSTEMS ENGINEERING OF AN ON-LINE DIAGNOSTIC SYSTEM

Jeffrey S Lin
The Johns Hopkins University Applied Physics Laboratory
Johns Hopkins Road
Laurel, Maryland 20723-6099

Abstract: On-line diagnostic systems primarily use data acquisition systems and expert systems to implement the diagnostic reasoning of a human expert. But many technologies must be integrated for the development of a successful system. These technologies include:

expert systems, data acquisition, and sensor validation,
data communications,
human/computer interfaces (HCI) and uncertainty management, and
system integration testing.

The success of the system will be diminished if any of these technologies, or tests of their implementation or integration, are not planned for from the beginning of the project. This paper reviews the lessons learned during the development and testing of the Air Compressor Diagnostic System.

Key Words: expert systems; monitoring; on-line diagnostics; sensor validation; systems integration; uncertainty.

INTRODUCTION: The Johns Hopkins University Applied Physics Laboratory (JHU/APL) recently completed and delivered the Air Compressor Diagnostic System (ACDS) to the U.S. Navy for the Integrated Diagnostics Demonstration (IDD). ACDS is an on-line monitoring and diagnostic system for reciprocating high- and low-pressure air compressors (HPAC and LPAC) [1].

The primary goal of automated diagnostic systems like IDD is to reduce machinery life-cycle costs by minimizing the costs of maintenance and repair operations. The savings result primarily from reducing the manpower needed to perform required tasks. A timely and accurate diagnosis of machinery condition prevents cascade failures from causing further equipment damage, and assures that the correct problem is addressed by the repair crew. A carefully constructed human/computer interface (HCI) helps insure that the diagnosis is understood and accepted by the user.

Other benefits can be expected from the application of automated diagnostics. If failing machines are identified more quickly and repaired, the operational availability and efficiency of the equipment will increase, improving shipboard operations and reducing energy costs. In addition, logistical tasks can be automated as part of the response to the detected failure.

The HPAC and LPAC versions of ACDS are two of several subsystems designed to diagnose various machines on a DD-963 class ship as part of the IDD. The subsystems communicate their diagnostic results to the IDD Core System, which integrates the results into a common interface for the user. The subsystems must first identify failing or failed machinery components, and then provide the user with the information necessary to respond to the failure.

The successful development of ACDS demonstrates the importance of systems engineering. The ultimate goal of systems engineering should be the satisfaction of the needs of the user. With this in mind, ACDS was developed to provide the U.S. Navy with a usable and useful tool for performing air compressor diagnostics.

DIAGNOSTIC METHODOLOGY: The data-acquisition and expert-system modules of a diagnostic system provide the core of the system functionality. Data acquisition is a mature technology and many good commercial-off-the-shelf (COTS) systems can be found for a new application. An aspect of data acquisition that is frequently ignored, however, is the validation of the sensor data [1,2]. Indeed, two diagnostic subsystems are generally required for each diagnostic system: one for the target machine, and one for the diagnostic system hardware itself. These both can be implemented in the same expert-system software.

An entire spectrum of COTS expert systems is available that offers widely varying capabilities. In general, the designer of an on-line diagnostic system should select a flexible, fully-capable expert system. Such an expert system includes a knowledge-representation capability, mathematical computation capability, and programming interfaces to external programs. Neuron Data's NEXPERT OBJECT™ package, which provides these capabilities, was used to develop the ACDS knowledge base.

Many software designers new to on-line diagnostics assume that diagnostic rules are simply an if-then construct that can be implemented in C or Pascal. This may be true for small diagnostic systems. As the number of diagnostic rules and the complexity of the diagnostic system increase, however, the more the software engineer will appreciate the functionality of a well developed knowledge-base-development environment.

The object-oriented nature of NEXPERT OBJECT assists greatly in the encapsulation of the representation of the world and the diagnostic rules. For instance, the ACDS knowledge base has a class called *component*. The class *component* has several attributes: faulted, severity, and confidence. Every air-compressor component that is diagnosed by ACDS is represented by an object of this class. Each of these objects inherits the attributes of the class *component*. Therefore the *oil_pump* object has several attributes. *oil_pump.faulted* is a Boolean hypothesis that can be TRUE, FALSE, UNKNOWN or NOTKNOWN. *oil_pump.severity* indicates a

warning or alarm level of the condition of the oil pump. *oil_pump.confidence* holds the confidence factor, low, moderate, or high, that *oil_pump.faulted* is TRUE.

Not all expert systems provide an UNKNOWN or NOTKNOWN value to a hypothesis. The distinction between these values is subtle, but important. A hypothesis has a value of UNKNOWN if the rule has not yet been evaluated. A hypothesis has a value of NOTKNOWN if the rule has been evaluated, but insufficient information exists to make a judgment. Examples of their use may help show their usefulness and difference.

ACDS loads only a subset of the rules for a given diagnosis, depending on how long the machine has been on or off. The value of the hypotheses of the rules not loaded are UNKNOWN. The following scenario illustrates the importance of this value.

Suppose that ACDS finds that the oil pump is faulted because the oil temperature is too high. Then, the air compressor, and with it the oil pump, happens to cycle off. The oil temperature will drop when the machine is off for a while. If the oil-temperature rule were loaded and evaluated at this time, the hypothesis that the oil pump is faulted would evaluate to FALSE, indicating that the pump is healthy. The pump was faulted, however, when the air compressor was on, so it is most likely still faulted when the machine turns off. To improve consistency, ACDS does not load this rule when the machine is off. The hypothesis that the oil pump is faulted will be UNKNOWN. The best guess for the health of the oil pump is the last known verified hypothesis, namely that the oil pump is faulted.

The NOTKNOWN value of a hypothesis is used when ACDS detects a sensor failure. If, during the sensor validation routines, a sensor value is found to be completely unreliable, its value is set to NOTKNOWN. The hypothesis of any rule that uses this sensor value then also evaluates to NOTKNOWN. ACDS informs the user of the sensor failure, and uses the last known verified hypothesis for those rules that are affected.

The object-oriented nature of NEXPERT OBJECT, through meta-slots, also allows actions to be defined when a specific data value changes. ACDS uses this feature extensively to keep data consistent within the knowledge base. All of the air-compressor components are grouped into air-compressor subsystems in a hierarchical fashion. For instance, in the HPAC ACDS, the 3rd stage inlet valve, discharge valve, relief valve, and rings comprise the 3rd stage subsystem. The five stage subsystems form the air subsystem. The air, lube/drive motor, cooling water, condensate drain monitor, and temperature monitor subsystems are parts of the overall HPAC system. This hierarchy is duplicated graphically at the HCI. The health of each system and subsystem depends on the health of the constituent subsystems at the lower level. ACDS updates the health of each subsystem through the meta-slots provided with the knowledge base.

Diagnostic-system designers who select weak expert system tools severely limit their ability to provide a robust, flexible, and useful product. Not all required functionality or features of an expert system may be known at the beginning of the development cycle. It can be cheaper to buy a fully-featured tool, and use only 30% of the capabilities than require 130% of the capabilities

of a less expensive tool. Procedural computer code can be written to compensate for deficiencies in the expert system, but that code must be written, tested, and maintained.

DATA COMMUNICATIONS: Unless the on-line diagnostic system resides with the target machine, some form of data communication will be necessary. Machinery environments may accumulate several diagnostic systems, using a variety of hardware and software that are tied together into a central computer. ACDS, for example, communicates data and diagnostic results to the IDD Core System for display to the user. The successful communication of information requires detailed specifications and early and frequent testing.

The specifications must include more than a hardware interface and a communications protocol. JHU/APL worked closely with the IDD Core System developer to define the timing, format, and content of the communications between ACDS and the Core System. Several tests of increasing complexity assured the successful communication. The first tests verified the hardware interface and communications protocol. Later tests verified the correct timing and successful communication of data. Most communication tests had actually been performed before the final message content and format were fixed.

One important lesson learned was to test every statement in the specification document at the earliest possible time. Some tests were forgotten at early tests for ACDS that required non-trivial software changes during later tests. Fortunately, we were able to make and test the changes without any delay in the schedule.

HUMAN/COMPUTER INTERFACE: An effective HCI for a diagnostic system must perform several tasks. First, all HCIs viewed by a given operator must be consistent. Second, the HCI must filter the information displayed to the user, allowing quick, clear communication of significant data or instructions. Additionally, an effective uncertainty-management technique must be used to qualify the information presented, so that the user understands the quality and reliability of the information.

The requirement for a consistent HCI can be met by either one common interface for all diagnostic systems, or a well defined specification and strict enforcement of that specification. Otherwise, the operator must be trained to use a new interface with each addition of a diagnostic subsystem. Little tricks or features that one diagnostic subsystem supplies may appear useful at first, but will eventually confuse the operator when it is not available in other subsystems. Apple Macintosh™ users long have enjoyed and extolled a consistent user interface, whereas users of IBM-compatible personal computers have only recently enjoyed the standard interface supplied by Microsoft Windows™.

The IDD imposed one common interface, which was supplied by the Core System, for all diagnostic subsystems. The result is transparent switching between diagnostic subsystems at the HCI. The user can learn to investigate a failure in one subsystem, and immediately transfer this knowledge to another subsystem.

JHU/APL worked with the Core System developer to define the user interface for the IDD. JHU/APL used cognitive models and task analysis to develop a user-centered approach to define the user-interface requirements for diagnostic systems [3,4].

The HCI requirements for an expert diagnostic-system such as ACDS are numerous. Air compressors are mechanically complex and, as a result, the diagnostic algorithms are sophisticated. An appropriate level of abstraction and representation of the system information is required. In addition, an HCI should provide:

an explanation and justification for a diagnosis and recommended action,
guidance for conducting additional tests and querying the system,
the ability, appropriately restricted, to alter decision limits of diagnostic rules, and
methods for obtaining access to diagnostic and repair procedures and documentation.

The HCI design must support the users, which in the case of ACDS are sailors with a wide range of knowledge and experience. JHU/APL built a prototype user interface with a hierarchical structure allowing multiple levels of interaction at the IDD user interface. Surface-level interactions allow the quick communication of problems and suggested remedies. Deeper-level interactions allow detailed investigation into the data and diagnostic algorithms.

Given the complexity of the air-compressor systems and the ACDS diagnostic algorithms, JHU/APL developed a model of diagnostic uncertainty [1] to assist the user in interpreting ACDS's diagnoses. The uncertainty in a diagnosis includes uncertainty associated with: the validity of the sensor data, the decision threshold limits, the efficacy of the rules, and confirmation or lack of confirmation by additional rules. The combination of these uncertainties is transparent to the user. The HCI only displays the resultant low, moderate, or high confidence in the final diagnosis. Additional resolution or detail in the level of confidence will only serve to distract or confuse the user. The HCI provides all data and explanations at the deeper-level interactions, for the user to further investigate and form his/her own opinions.

JHU/APL recommended an HCI implementation and test plan, that included prototype walk-throughs, user feedback, and usability studies. Although funding limitations prevented the plan from being executed for IDD, the execution of such a plan will improve and validate the effectiveness of any diagnostic system HCI.

INTEGRATION TESTING: Lastly, system-integration testing is a vital part of the development of all complex systems, including on-line diagnostic systems. System integration was required at two levels for the IDD. At the subsystem level, the data acquisition, diagnostic, and communication hardware and software systems were integrated and tested. At the IDD system level, each subsystem had to be integrated with the Core System and fully tested. For ACDS, these two levels of integration testing were performed concurrently.

The first ACDS functionality tested was the communications. Simulated data files were transmitted from one computer to another using the communication protocol specified by the Core-System developer. These tests were followed by tests with a Core-System simulator

written by the Core-System developer. Several tests were performed, each exercising successively larger portions of the communications capability. All test procedures performed at one integration test with the Core System were repeated during later tests, to assure that no functionality was lost.

Concurrently, the HPAC diagnostic knowledge-base was tested against existing data taken from a fault-insertion test program. This data does not exhaustively cover the faults that ACDS diagnoses, and had to be supplemented with artificially-generated data. The Dresser-Rand Company, which supplied the diagnostic expertise for ACDS, used simulation codes to generate the additional data. No existing fault-insertion test data is available for the LPAC, therefore Dresser-Rand Company supplied similar artificial data to test all of the rules in the LPAC ACDS. The sensor-validation rules also were exhaustively tested by simulated data.

Once the knowledge base was verified in the NEXPERT OBJECT development environment, it was integrated with the ACDS main program, which already was proven to communicate with the Core System. The main program is written in C and accesses the knowledge base through an application programmers interface with C function calls. Simulated sensor-data was stored in data files, read in by the C program and passed to the knowledge base. The results of the knowledge-base execution were then read back into the C main program, and transmitted to another computer. Once this capability was demonstrated at JHU/APL, we again tested the integration of ACDS with the Core System.

The next step was to perform the data acquisition to obtain real-time sensor data. The data is acquired through C function calls to a library of routines supplied by the vendor of the data-acquisition hardware. First, the data acquired was verified independent of the rest of ACDS. Next, ACDS sent the data to the Core System, where it was verified as accurate. Due to cost constraints, realistic sensor-values could not be simultaneously simulated across all data acquisition channels inputs. We could not, therefore, control the diagnostic output of ACDS. Since the arbitrary data of open data-acquisition channels violate many sensor validation checks, only sensor failures are detected. Therefore, for subsequent tests with the Core System, all data that was acquired was overwritten by data from test files before being sent to the knowledge base.

The full diagnostic functionality of ACDS was demonstrated during an integration test with the Core System. Concurrently, the full shipboard-ready hardware/software integration of ACDS was verified independent of the Core System.

The last level of testing before shipboard installation was originally planned to be a land-based test. All of the subsystems were to be installed on the targeted machinery and connected to the Core System. The land-based test would have served as an excellent validation of IDD. The cost estimates for the land-based test, however, soon became prohibitive.

To replace the land-based tests, an IDD simulation/stimulation system was planned to demonstrate sensor-to-HCI connectivity. A simulation/stimulation test would verify each subsystem functionality, and the successful integration of the subsystems and the Core System.

Unfortunately, program funding limitations prevented the completion of this final level of integration testing.

CONCLUSIONS: Developing a successful on-line diagnostic system requires early attention to the performance requirements as well as the testing requirements. Software tools should be selected to allow flexibility in the implementation of the knowledge base. Integration, verification, and validation tests are time consuming and expensive but necessary. In an operational environment where many separate diagnostic systems must coexist, these test are imperative. Without these tests, the user cannot be insured of getting a system that solves the right problem, works with installed systems, and presents the right information to the user in an understandable, consistent format.

ACKNOWLEDGMENTS: The author wishes to acknowledge Dr. Augie DiGiovanni and Mr. Mike Good of the Carderock Division of the Naval Surface Warfare Center for their support and assistance with the many technical issues in this work. The author also acknowledges Mr. Scott Delmotte of the Dresser-Rand Company for providing his expertise in compressor diagnostics.

REFERENCES:

[1] J. S. Lin, S. Delmotte, "On-Line Diagnostics of Reciprocating Multi-Stage Air Compressors," Advanced Materials and Process Technology for Mechanical Failure Prevention, 48th Meeting of the MFPG, pp. 369-378, 1994.

[2] J. S. Lin, C. L. Resch, J. V. Palmer, "Sensor Validation for On-Line Diagnostics," Proceedings of the 40th International Instrumentation Symposium, pp. 519-528, May 1994.

[3] B. G. Coury, J. S. Lin, F. B. Weiskopf, "System Representations, Cognitive Modeling and the Design of User Interfaces for an Expert Diagnostic System," Proceedings of the IEEE/SMC'93 Conference, pp. 26-31, October 1993.

[4] B. G. Coury, R. D. Semmel, F. B. Weiskopf, J. Jackson, "Attention Direction in Uncertainty Management," in M. Mouloua and R. Parasuraman (Eds.), Human Performance in Automated Systems: Current Research and Trends, (Hillsdale, NJ: Lawrence Erlbaum), pp. 270-276, 1994.